WME Site Organization and Customization Support

David Chiu
Institute for Computational Mathematics
Department of Computer Science
Kent State University, OH
dchiu@cs.kent.edu

ABSTRACT

Minimizing work effort in deployment of mathematics educational Web pages calls for an encapsulation of lesson material into self-contained Topic Modules. Methods for creating interoperable content pages, supporting automatic page generation, enabling page configuration, and allowing user customization are presented.

INTRODUCTION

A Web-based Mathematics Education (WME) [1] Website was deployed to a group of 7th grade students at Kimpton Middle School (Munroe Falls, Ohio). This pilot Website offers online activities to aid mathematics lectures. The WME site requirements and our initial findings for this project are reported in [2].

At the time of developing the pilot Website, the parallel development of the Mathematics Education Markup Language (MeML) [3] and the MeML processor [3] were still under investigation. In fact, their progress was reliant on the results produced by the pilot. This feedback is used to assess MeML and Woodpecker's feasibility in an arrangement to amend or deprecate its existing elements which may turn out to be extemporaneous from a lack of experimenting in realistic situations. Fortunately, because the pilot project was initiated during the school year, feedback from both students and faculty were instantaneously available. With this steady stream of feedback, we compiled a general set of requirements for Topic Modules:

- *A Means for Efficiency* --- A way to maintain quick and persistent progress in the development of the pilot Website is necessary in order to stay on pace and synchronized with the ongoing classes at Kimpton Middle School.
- *Topic Modules* --- Mathematics topics (percentages, proportions, fractions, etc) are viewed as *modules* that can be loaded/unloaded from any Website built on the WME Framework. These self-contained Topic Modules (TMs) consist of a set of Topic Lesson Pages (TLPs) which contains mathematical lessons, exercises, activities, etc, that are designed to complement the physical lectures given in a classroom.
- *Interoperability and Relocatability* --- A means to seamlessly incorporate or relocate mathematical TMs into WME websites is quintessential to the WME Framework [4].
- *Customization* --- Support for custom configurability is certainly necessary in the likelihood that a teacher chooses to customize lessons within TMs to better fit his/her method for teaching. It also allows for the same TM to be used by different teachers and classes within schools to maximize code (or in this case, lesson) reuse.

The pilot site employed many areas where hard-coding fulfilled much of the implementations of the above requirements. For example, our preliminary TMs and TLPs were designed specifically for the requirements of Kimpton's teachers and courses with content that are customized fully to their needs -- far from WME's ultimate goal. Yet, hard-coding provided us with enough information to explore its practicality and envision the design of pragmatic tools to support a generalized method for WME support.

WME SITE CONFIGURATION

For a Website to be WME compatible, a series of XML files, *WME Site Configuration Files*, must be existent on the Web server. These files contain Website-wide data such as a list of installed TMs and information regarding the school's teachers, students, and courses. *WME Site Configuration Files* represent an intricate part of WME site, as they provide information to TMs in a way to define the relationships needed for supporting interoperation and customizability. In fact, TMs shall be designed in such a way that it always anticipates *WME Site Configuration Files* to be readily available. The technical aspects of these files are discussed in [8], which we will later revisit.

INSIDE TOPIC MODULES

Most people that we envision using WME are not necessarily computer scientists. In light of this, WME aims to provide a trivial environment to deploy mathematics educational material onto the Web. In an attempt to achieve this goal, we lessen the amount of work needed from a potential user's point of view (a school administrator or a math teacher). This basically involves minimizing the amount of effort needed to not only load or unload TMs into WME sites, but to see that they are immediately operable or relinquished. Provisions for this intuitive site management implicate higher file complexities in TM designs to assure self-sufficiency. In other words, TMs must be supplied with not only lesson Web pages, but also its *own* set of configuration files bounded onto the assumption of the presence for *WME Site Configuration Files* -- its only dependency. One might be inclined at this point to speculate as to why *any* link back to the WME site is necessary. Clarity may arrive with later sections on *relationships*.

Inclusive within every TM are the following files [8]:

- *Template Files* --- The basic structural content defining the fundamental "look and feel" for every TLP within the module.
- *Content Files* --- The text, illustrations, demos, etc that needed to convey mathematics lessons. A caveat is that *one* TLP may contain *multiple* content files.
- *WME Service Files* --- The optional inclusion of these files offers local WME Service procedure calls. These might contain simple on-the-fly calculations and page manipulations.
- *Assessment Files* --- A set of two files containing questions and feedback. The question file contains teacher-defined questions for specific TLPs. The response file contains the student feedback to these questions.
- *Configuration Files* --- The set of XML files that marks the only dependency link to the WME site.

The final item in the list above provides the foundational trestle in supporting interoperation and customization. These files can be seen as an inner-TM database maintaining such information as which distinctive files are needed for the creation of TLPs, a guide for mapping customization values to their respective pages, etc. The individual files in the set of *Configuration Files* are identified here:

- *tm_conf.xml* --- Contains static TM information such as its version, the author, etc. Also defines each TLP's file structure, discussed in the following section.
- *tm_customization.xml* --- Contains manipulative parametric values for TLPs.
- *tm_map.xml* --- Defines the relationship values from the TM to the WME site. Here the only place where *alien* WME site information is introduced *into* the TM (teacher or course information specific to the site).

From Figure 1, every TLP is the collective assembly of the template, content files, and WME Service files, with selective data drawn from the customization and assessment files. Next, we discuss this architecture.

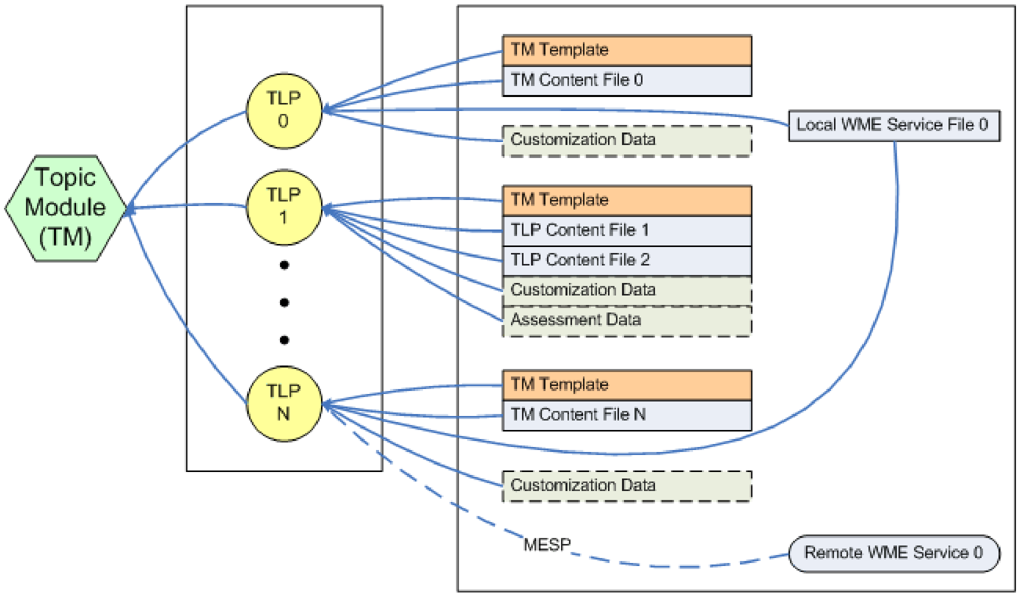

FIGURE 1. A Topic Module Architecture

THE TM ARCHITECTURE

For clarity, we will discuss the architecture given by Figure 1 in detail in respect to its integral parts starting from the primary level component, a single TM. The organization of a TM is in essence a makeup of multiple TLPs. Aside, "TLP" is actually rather deceptive in its name. A single TLP is in fact a skeleton that contains nothing but *file-include* statements -- basically an acting container for the inclusion for miscellaneous files that ultimately define its logical existence. These definitive files exist in the tertiary level of our figure.

From a fleeting glance, it is easy to conclude that the *Template* is included with every TLP skeleton. This is no surprise. After all, templates are by nature a global entity in order to efficiently spread its elements to onto its employers (in our case, TLPs). Thus, one modification to the global template affects all TLPs. The template allows for simple and quick deployment for consistent results -- the basis of adding design and style for all TLPs.

Dissecting further, it seems that tertiary level components, such as *Content Files*, instinctively spider across the platform to create the intuitive illusion of TLPs that are neatly juxtaposed in the secondary level. Later, we will explore *how* the tertiary constituents are mapped in respect. But first, we delve into *why* the ostensible singularity of TLPs must be divided into smaller parts.

After the inclusion of the design template, the TLP skeleton appends its content files. The skeleton structure of TLPs is to offer robustness by exploiting dynamic content generation. For instance, TeacherA might use some TLP, *add_fractions.meml*, with some fraction addition content that is only relevant to him/her. In contrast, TeacherB's rendition might have a slightly different organization or entirely dissimilar content. However, in either case, the same file entity, *add_fractions.meml*, is displayed on the browser -- just different versions per its teacher.

Notice the disparities between TLP0 and TLP1 in Figure 1. TLP1 includes multiple content files -- *but why*? While the impetus behind this mock TM example might be elusive, reasons for supporting multiple content files have been the catalyst for recent discussions of allowing Administrative Page Control -- in particular to "page blocking". In fact, page blocking is not a novel concept. *Recall back to grade school when teachers can force an entire class to focus on a distinctive section of their light-projected lecture notes by allowing some pieces of strategically positioned papers to offer its opacity on the projected image. The resulting image projection, of course, is the focal point.* By splitting a full set of "electronic" lecture notes into sections placed in multiple content files, TMs achieve similar effects. *While lecturing during a WME-enabled class session, a teacher can simultaneously control which page sections should be hidden or displayed at any particular point in time.* How these sections shall be broken up (in other words, how many content files per TLP), is, of course appropriated by the teacher as they see fit for their lecture.

This *content splitting* can also useful for lesson sharing. For instance, TeacherA creates some content file to his/her liking. In a different class, TeacherB can use TeacherA's provisions by opting for an exchange or inclusion of TeacherA's newly devised lecture. Again, this is nothing new or exciting as file inclusion has already been widely used by the industry to offer this type of dynamic content presentation. *But the advantage here is that WME interoperability not only guarantees dynamic content display, but also the immediate recognition and operability of the "newly" generated TLP without user intervention.*

But even if TeacherA devised his/her lecture content, it would infer TeacherA be knowledgeable with the TM architecture, or perhaps even HTML and other Web technologies -- *not exactly a strong assumption for us to make, but we nonetheless leave this option open*. Instead, a less drastic approach for manipulating TLP content is done through configuring page customization values.

We trace back to the example posed by Figure 1. Subsequently after inserting content file(s), the TLP skeleton summons for page customization data. Seen in the figure as rectangular boxes bearing a dotted-line border, customization data are not to be confused with actual files. While all customization values originate from a single file, *tm_customization.xml*, the difference here is that the TLP skeleton includes only those custom data values specified for the TLP in question, and not the entire customization file. These custom values populate predefined placeholders in the template or TLP content files. Again, arranging the need for per-teacher or per-class customization, TLPs rely on the values defined in *tm_customization.xml* to give itself not only substance, but the lesson conveyed to the teacher's liking.

Alluding from prior discussion, *WME Service Files* are actually incorporated into the skeleton before content files. Often, *Content Files* contain procedure calls to achieve some form of student interaction such as responding to a mouse click in an activity or checking students' answers in exercise questions. These WME Services [1] can be issued locally via a Web program or invoked remotely through the Mathematics Education Service Protocol (MESP) [1]. Because there are two types of WME services, there exist two methods of invocation:

- *Local WME Service Invocation* --- *Local* WME service invocation can be achieved by using any arbitrary function call -- as long as the function is defined and programmed locally on the server, inside *WME Service Files*.

- *Remote WME Service Invocation* --- In order to invoke a *remote* WME service (i.e. the procedure exists elsewhere off the invoking server), the URL of the function call is appropriated. Function parameters and result data are placed directly inside page markup, and are passed back to the WME server specified by the URL via MESP.

Indeed, it is both likely and conceivable that various content files might share the need for identical local WME Service invocations. Thus, the TLP skeleton might include multiple *WME Service Files*. TLP0 and TLPN in Figure 1 illustrate this behavior. Also notice the additional link between TLPN and a remote WME Service, resident on some other WME server, communicated through MESP.

Finally, the optional assessment data is appended to the TLP skeleton. Note that by now, our TLP skeleton has gathered the body of its substance: template for style definition, WME service files to handle local procedure calls, and lesson content including text, graphics, exercises, etc. In the rear end of the lesson page, the skeleton affixes any student assessment questions. Its late positioning is, nonetheless, well-exceeded by its practical importance: *allowing teachers to assess students' understanding of the mathematics lesson topic depicted in the TLP*. Like that for customization data, the TLP skeleton resists the inclusion of the entire assessment question file, *tm_assessment.xml*. Only relevant assessment data is incorporated into the TLP. On the other hand, the inclusion process is more complex than the inclusion of customization values, which simply involves populating page variables. A search query is run to uncover all assessment questions related to the TLP in question. This list of questions is then further refined depending on which course is in session and who is teaching it. The final question list is displayed at the end of the page. The students' response to these assessment questions must also be communicated and accommodated. Their response is placed into the question file's counterpart, student response file: *tm_response.xml*.

At this point, we are left partially clear as to how TLPs are dynamically generated. The above describes the TM architecture and a TLP's general organization in accordance to a skeleton for dynamic file incorporation. Yet, the governing rules for not only the maintenance of this crucial organization, but also the management of page customization and interoperation lie deeply within the briefly described *TM Configuration Files*.

MANAGING RELATIONSHIPS

By observing the above physical model, we are drawn to the inference that TMs are, for the most part, self-reliant. But a bit of contemplation leads one to deliberate a TM's efficacy without *some* information from its employers. *How useful are a TM and the lessons that it attempts to convey without any intuitive direction for its usage? How would a TM be able to offer its per-class and per-teacher customization services when both entities are indigenous to the system, and not to TMs?* To answer this, a very obvious link between the TM and the WME Website must be established. Next, we discuss support for the data link in both directions, but it should be mentioned that we are *not* defining a novel approach protocol. What we are offering is a model for containing persistent and relational data involving TMs. The link we refer to is a simple query for this data, implemented by some XML parsing scripts or, as mentioned before, databases and SQL.

Forward Association

Forward association refers to Website-to-TM data communications. But to understand this direction, we must again consider the *WME Site Configuration Files*. Reiteratively, these files are the site-wide accommodations for the institutionally relevant data:

- *teacher_conf*.xml --- A list of teachers and their information (unique user ID, full name, login, etc).
- *student_conf.xml* --- A list of students and their information (likewise to those given by teachers).
- *course_conf.xml* --- A list of available courses and their information (unique course ID, name, time, location, teacher, etc). Particularly of interest is a sub-list of students who are enrolled.
- *tm_component.xml* --- The list of TMs that are docked and ready for use (unique TM ID, location).

Although the first three files irrefutably contain rather significant data, it is the latter file that is key simply because without it, the site would be ignorant of any TM hosted on the site. It should be quite noticeable that all of the above components are identified by some unique ID. Equipped with a means for identification, the *tm_component.xml* file is able to provide the server with the loaded TMs' locations, ultimately establishing a means for the site to reference each TM, both as the source (for reads) and destination (for writes). Thus, a forward link (to read and write to and from the WME site) is enabled. Figure 2 offers the simple concept behind this model.

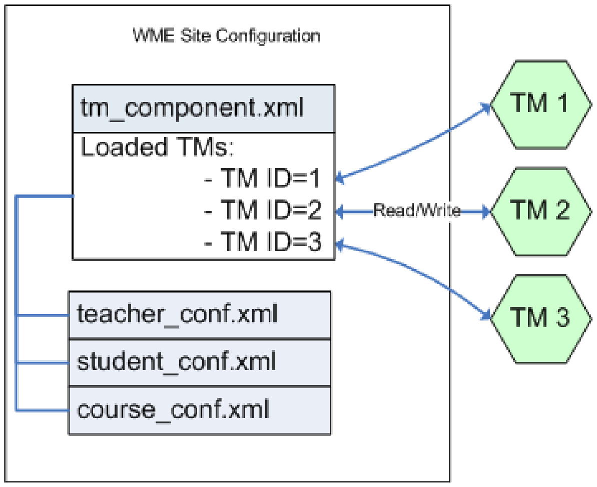

FIGURE 2. The tm_component.xml File Enables Forward Association.

With the forward link in place, values stored in teacher, student, and course configuration files are now outfitted with the *direction* mapped by *tm_component.xml*. This association allows for implementation of such interfaces as user login and recognition which enables a slew of applications (Administrative Page Control, for one). Whereas the function for a forward relationship is realized, the purpose behind a reverse association is equally tangible.

Reverse Association

In the opposing direction, reverse association deals with those data communications from TM to Website. A reverse connection is necessary for TMs to acknowledge for whom they are working (those course, teacher, student information defined previously on the site). Similar to that of forward associations, the data's source is given by the main configuration file -- in the case for TMs, *tm_conf.xml*. A major difference, however, is that TMs are not allowed to write into *WME Configuration Files*, such that the reverse link is only enabled for read operations. It makes sense, after all, since TMs are a supposedly workforce controlled by the site and not the other way around.

A TM's main configuration file, *tm_conf.xml*, contains the physical location of the *WME Site Configuration Files*, thus acting as the sole tunnel of data queries, as depicted in Figure 3. Data reception, however, is asserted directly into the mapping file, *tm_map.xml*. We pointed out earlier that *tm_map.xml* contains those values retrieved from the server. It is also the interface between TLPs and their custom page variables and assessment data. Because customization and assessment data are per-class and per-teacher, a dynamically generated TLP must know which class and teacher it is currently being used by. Every time a TLP is loaded, it must query for relevant customization and assessment values. *A dynamic TLP would not be so dynamic if this information is static!* The query is issued to *tm_map.xml*, which asks for the location of server configuration files from *tm_conf.xml*, and is finally invoked on the appropriate files (of course, *which* file to run the query for depends on the information that the TLP is asking). *Tm_map.xml* then matches the retrieved data with the abundant set of values in *tm_customization.xml* and *tm_assessment.xml* and depending on these values (teacher ID and course ID, perhaps), filters the set of customization and assessment data into those used by the TLP in question, the course, and the teacher in which issued the invocation.

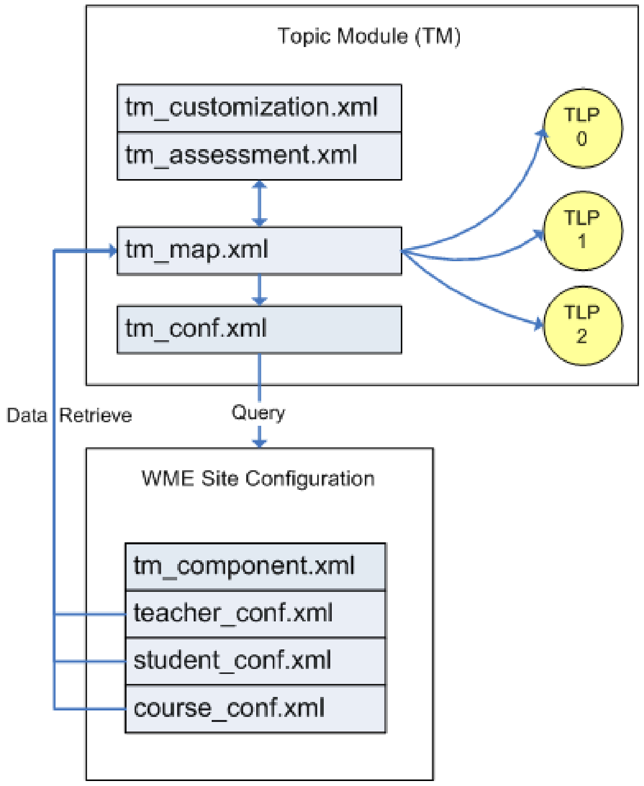

FIGURE 3. The Reverse Association.

CONCLUSION

Up to now one can argue that TMs, with some minor adjustments, can probably be applied to *any* general on-Web education. *Indeed, why narrow its applications to only mathematics education?* More willingly than refuting this argument are we to support it. If we can offer TMs the capabilities for mathematics presentation and input, its applications can be generalized. It could possibly support *any* subject that current Web technologies can sustain including science disciplines where mathematics will certainly be applied. In retrospect, this seems to relate our works of TMs to the heap of unprecedented online Learning Management Systems such as e-Learning [9] and WebCT [10]. But despite potential for general education, let us not allow our initial focus of mathematics go awry. So, the question now becomes: *How do TMs support the mathematical nature of education?*

In WME's implementation, TMs are, in essence, composed of TLPs which will eventually be written in MeML. Mathematics is supported within MeML in many ways. Because it supports a mixing of itself with MathML, mathematics display is achieved. On the other hand, mathematical computation is handled through WME Service invocations to services from as simple as pre-programmed algorithms to interfacing with such computer algebra systems as Maxima (formerly MACSYMA) [11], Maple [12], and Mathematica [13]. But what do MeML's mathematical components have anything to do with the design of TMs? *Well, everything.* TM's notion for customization implies that mathematical expressions can vary due to the teacher's or course's discretion. The same dynamic behavior applies to parameters for a computational function perhaps, or mathematical notation (infix, postfix, natural) and their conversions, etc. To not only support mathematics (via MeML and MathML), but also allow these mathematical constituents to be customizable and interoperable is our definitive goal.

FUTURE WORK

Our persistent effort to introduce WME into classrooms is continuous. While its current state is a mostly product hard-coding, the migration into to the standard model for Topic Modules has already begun. This in itself will be assessed for robustness, scalability, and of course, interoperability.

The technical feedback received from the pilot Website is currently being assessed in order to further refine MeML for handling TM requirements. The MeML processor and MeML authoring tool must also undergo reevaluation to reflect the updated specifications. Additionally, subscribing MathChat [7] to the Website might implicate TMs in ways we have not considered. A way to link topics discussed in chat to TMs and their TLPs may be beneficial, but whether or not accommodations for MathChat have to be made is still being addressed.

ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation under Grant No. 0201772. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Wang, P.S., Kajler, N., Zhou, Y., et al, “WME: Towards a Web for Mathematics Education”, *Proceedings of ISSAC*, ACM Press, Philadelphia PA, August 2003, pp. 258-256.

[2] Mikusa, M., Wang, P.S., Chiu, D., et al, "Web-based Mathematics Education Pilot Project", *Proceedings of ITE*, McGraw-Hill, Elizabethtown PA, September 2004 (to appear).

[3] Wang, P.S., Zhou, Y., Zou, X., “Web-based Mathematics Education: MeML Design and Implementation”, *Proceedings of IEEE/ITCC'2004*, IEEE, Las Vegas NV, April 2004, pp. 169-175.

[4] Wang, P.S., Kajler, N., Zhou, Y., et al, "Initial Design of a Web-based Mathematics Education Framework", *Proceedings of IAMC'2002*, Lille France.

[5] Max Froumentin,Team Contact for the Math Working Group. *MathML*. www.w3c.org/Math.

[6] The World Wide Web Consortium (W3C). http://www.w3c.org.

[7] Chiu, D., "Web-based Mathematics Education with MathChat", *Proceedings of IEEE/ITCC'2004*, IEEE, Las Vegas NV, April 2004, pp. 709-717.

[8] Chiu, D., "Design and Implementation of the MathChat Protocol and a Model Educational Website for the WME Framework", *Master's Thesis*, Department of Computer Science, Kent State University (to appear).

[9] e-Learning Online Learning Management Systems. http://www.elementk.com.

[10] WebCT. http://www.webct.com.

[11] Pavelle, R., Wang, P.S, "MACSYMA from F to G", *Journal of Symbolic Computation*, Academic Press, vol. 1, 1985, pp. 69-100.

[12] Maple. http://www.maplesoft.com.

[13] Mathematica. http://www.wolfram.com.